Flight Price Prediction Using Streamlit

Aakash Singhal
Northeastern University
Boston, USA
singhal.aak@northeastern.edu
Gihub Link : https://github.com/aakutka/Flight-Price-Prediction/tree/main

***Abstract*—This project focuses on developing a machine learning model to predict flight prices based on various features such as airline, date of journey, departure and arrival times, total stops, and additional information. The dataset underwent extensive preprocessing, including data cleaning, handling missing values, and outlier treatment, to ensure data quality. Feature engineering was employed to extract relevant information from date and time columns, encode categorical variables, and normalize numerical features. A series of machine learning algorithms were then trained on the processed data to accurately predict flight prices. The final model aims to provide travelers with an accurate estimate of flight costs, enabling more informed decision-making. The project's results highlight the importance of data preprocessing and feature engineering in building robust predictive models.**

***Index Terms*—Linear Regression, Exploratory Data Analysis, Feature Engineering**

I. Introduction

In the modern travel industry, flight prices can fluctuate significantly due to a multitude of factors such as airline policies, seasonal demand, time of booking, and the number of stops on a given route. These fluctuations make it challenging for travelers to predict the best time to purchase tickets at an affordable price. Consequently, the ability to forecast flight prices accurately has become increasingly valuable for both consumers and service providers.

This project aims to develop a machine learning model that predicts the price of airline tickets based on various attributes. By leveraging historical flight data, the model seeks to identify patterns and correlations that influence pricing. The dataset used in this project includes features such as the airline, date of journey, departure and arrival times, total stops, and additional information provided by the airline.

The process began with an extensive exploratory data analysis (EDA) to understand the underlying structure and characteristics of the dataset. This step was crucial in identifying patterns, outliers, and relationships between the features that could impact the model's performance. Following EDA, the dataset was meticulously cleaned and preprocessed, addressing issues like missing values, inconsistent data types, and outliers. Feature engineering was then employed to create new, more informative variables from existing data, such as extracting day, month, and year from the date of journey or transforming categorical variables into numerical formats suitable for modeling.

The core of this project revolves around building and evaluating multiple machine learning models, ranging from basic linear regression to more complex ensemble methods like Random Forests and Gradient Boosting Machines. Each model was trained and tested on the processed dataset to determine its effectiveness in predicting flight prices. The models were evaluated based on key performance metrics such as mean absolute error (MAE) and root mean square error (RMSE), with the best-performing model selected for final deployment.

The ultimate goal of this project is not only to predict flight prices with high accuracy but also to provide insights into the factors that most significantly influence ticket costs. These insights can be valuable for travelers looking to optimize their flight booking strategies, as well as for airlines aiming to refine their pricing models.

In the following sections, we will delve deeper into the methodology used, the results obtained from various models, and the conclusions drawn from this analysis. The report will also discuss potential improvements and future directions for this project, including the integration of additional features or the application of more advanced machine learning techniques.

Fig. 1. Flight

II. Steps for the Implementation

The dataset analyzed in this project contains information about flight prices and various associated attributes. The dataset comprises columns such as Airline, DateofJourney, Source, Destination, Route, DepTime, ArrivalTime, Duration, TotalStops, AdditionalInfo, and Price. The primary objective of this data cleaning process was to prepare the data for

subsequent analysis by addressing issues such as missing values, incorrect data formats, and irrelevant features.

A. Data Preprocessing and Cleaning

Loading and Exploring the Dataset The initial step in the data cleaning process involved loading the dataset using the Pandas library. The dataset, named flightprice dataset, was read into a DataFrame by pandas, which provided a tabular representation of the data for easy manipulation and analysis. The dataset's first few rows were inspected to understand its structure and the nature of the data contained within it. This inspection revealed potential areas for cleaning, including inconsistent data formats and the presence of missing values. Handling Missing Values One of the most common issues in raw datasets is the presence of missing values. In this dataset, the TotalStops and Price columns were of particular interest. The approach to handling missing values involved: Identifying Missing Values: The dataset was examined to identify columns with missing values, and the percentage of missing values was calculated for each column. Imputing Missing Values: For columns where missing values were not significant (i.e., less than 10 percent of the data), median imputation was employed to fill the gaps. For columns with more substantial missing values, a decision was made to either drop these rows or fill them with meaningful substitutes, such as the most frequent value. Data Type Conversion The dataset contained several columns with incorrect data types, notably date and time columns, which were initially read as strings. Correcting these data types was crucial for accurate analysis: Date Conversion: The DateofJourney column was converted from string format to the datetime format to facilitate time-based analyses and calculations. Time Conversion: Similarly, DepTime and ArrivalTime columns were converted into a more suitable time format. Here in the dataset the format is in where it is mentioned in hours and minutes. since as open dataset all the values will get difficult to interpret differently in hours and minutes. So it is decide to convert the duration time in minutes to make the data easier for the models to interpret easily.

Postprocessing results: Below is the Image of the raw dataset :

Below is the Image of the Post processing of the dataset:

B. Exploratory Data Analysis(EDA)

Exploratory Data Analysis (EDA) is a crucial phase in any data science project, providing insights into the dataset's underlying patterns, distributions, and relationships between variables. The dataset in this analysis comprises attributes related to flight prices, including airline, dateofjourney, source, destination, deptime, arrivaltime, duration, totalstops, additionalinfo, and price. This summary outlines the EDA techniques used, the rationale behind each step, and the insights gained from the analysis. 1. Initial Data Loading and Inspection Tools Used: Pandas for data manipulation. Why: To understand the dataset's structure, identify missing values, and get an overview of the data types and initial data distributions. What

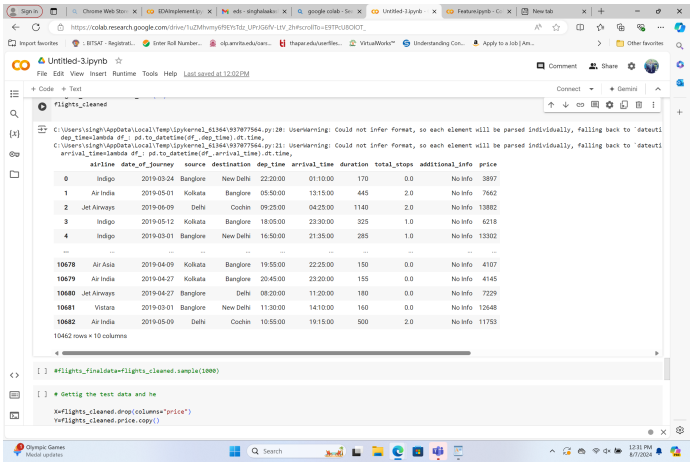

Fig. 2. Dataset Format

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302

Fig. 3. Preprocessing format

was done: The dataset was loaded into a Pandas DataFrame to facilitate easy manipulation and analysis. Initial inspection of the dataset included viewing the first few rows to get a sense of the data's structure and content.Different Types of operations has been done to get the better details such as univariate , bivariate, multivariate analysis has been done to get the better clarity of the datas.Methods such as distribution of individual features and identify any unusual patterns or outliers. What was done: Histogram and Density Plots is Used to visualize the distribution of price. Box Plots: Employed to detect outliers in numerical features, particularly focusing on price and duration.Performed the Pair Plots to understand different features at once on the output.

Performed the various Hypothesis testing to check the performance of the input features on the ouput features. ANOVA is a statistical method used to compare the means of three or more independent groups to determine if there are statistically significant differences between them. It helps to understand if any of the group means are different from the others, which

	airline	date_of_journey	source	destination	dep_time	arrival_time	duration	total_stops	additional_info	price
0	Indigo	2019-03-24	Banglore	New Delhi	22:20:00	01:10:00	170	0.0	No Info	3897
1	Air India	2019-05-01	Kolkata	Banglore	05:50:00	13:15:00	445	2.0	No Info	7662
2	Jet Airways	2019-06-09	Delhi	Cochin	09:25:00	04:25:00	1140	2.0	No Info	13882
3	Indigo	2019-05-12	Kolkata	Banglore	18:05:00	23:30:00	325	1.0	No Info	6218
4	Indigo	2019-03-01	Banglore	New Delhi	16:50:00	21:35:00	285	1.0	No Info	13302
...	...	...	...	...	...	...	...	...	...	...
10678	Air Asia	2019-04-09	Kolkata	Banglore	19:55:00	22:25:00	150	0.0	No Info	4107
10679	Air India	2019-04-27	Kolkata	Banglore	20:45:00	23:20:00	155	0.0	No Info	4145
10680	Jet Airways	2019-04-27	Banglore	Delhi	08:20:00	11:20:00	180	0.0	No Info	7229
10681	Vistara	2019-03-01	Banglore	New Delhi	11:30:00	14:10:00	160	0.0	No Info	12648
10682	Air India	2019-05-09	Delhi	Cochin	10:55:00	19:15:00	500	2.0	No Info	11753

Fig. 4. Postprocessing format

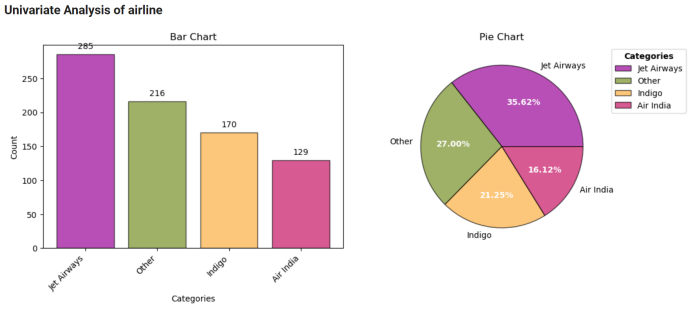

Fig. 5. Univariate Analysis

would indicate that the factor being tested has an effect.The Kruskal-Wallis test is a non-parametric alternative to ANOVA, used when the assumptions of ANOVA (such as normality and homogeneity of variances) are not met. It compares the medians of three or more independent groups to determine if there are statistically significant differences between them. Shown here is the screenshot summary of the methods used between the airline and the price of the flight

Hypothesis Test for Association between price and airline

ANOVA Test

```
- Significance Level   : 5.0%
- Null Hypothesis      : The groups have similar population mean
- Alternate Hypothesis : The groups don't have similar population mean
- Test Statistic       : 571.7959606959738
- p-value              : 0.0
- Since p-value is less than 0.05, we Reject the Null Hypothesis at 5.0% sig
- CONCLUSION: The variables price and airline are associated to each other
```

Kruskal-Wallis Test

```
- Significance Level   : 5.0%
- Null Hypothesis      : The groups have similar population median
- Alternate Hypothesis : The groups don't have similar population median
- Test Statistic       : 3856.6408669363445
- p-value              : 0.0
- Since p-value is less than 0.05, we Reject the Null Hypothesis at 5.0% sig
- CONCLUSION: The variables price and airline are associated to each other
```

Fig. 6. Hypothesis Testing

Here performed the numerical summary on the duration of flight column to check whether these are following normality rule,presence of outliers,performing the testing Shapiro Wilk test and the Anderson test to check the normality testing.

C. Feature Engineering

Feature engineering is a critical process in preparing a dataset for machine learning models. It involves transforming raw data into features that better represent the underlying problem to the predictive models, ultimately improving the model's performance.

In this project, feature engineering was systematically applied to enhance the dataset for predicting flight prices. The key steps involved in this process include: Encoding Categorical Variables One-Hot Encoding,Categorical features such as Airline, Source, Destination, and Additional Info were transformed into numerical format using one-hot encoding. This

Measure of Spread:

	value
var	259397.346527
std	509.310658
IQR	761.250000
mad	375.000000
coef_variance	0.781017

Skewness and Kurtosis:

	value
skewness	0.860837
kurtosis	-0.101528

Hypothesis Testing for Normality:

Shapiro-Wilk Test:

```
Significance Level   : 0.05
Null Hypothesis      : The data is normally distributed
Alternate Hypothesis : The data is not normally distributed
p-value              : 2.5171841510787788e-23
Test Statistic       : 0.8910248875617981
- Since p-value is less than alpha (0.05), we Reject the Null Hypothesis at 5.0% significance level
- CONCLUSION: We conclude that the data sample is not normally distributed
```

Anderson-Darling Test:

Fig. 7. Normality Testing

method created binary columns for each category, allowing the model to interpret these categorical variables effectively. Date-Time Features: The DateofJourney, DepTime, and ArrivalTime were parsed to extract meaningful components such as Day, Month, Year, Hour, and Minute. These new features helped capture temporal patterns in the data, such as seasonal trends or time-of-day effects on flight prices.Standard Scaling: Continuous features like Price and Duration were standardized to have a mean of 0 and a standard deviation of 1. This scaling was essential for algorithms sensitive to the magnitude of the input features, ensuring that all features contributed equally to the model.Winsorization: Outliers in the Price and Duration columns were treated using Winsorization, which caps extreme values to reduce their impact on the model. This step helped in stabilizing the variance and making the model less sensitive to outliers.

	air__airline_Air India	air__airline_Indigo	air__airline_Jet Airways	air__airline_M C
0	1.0	0.0	0.0	
1	0.0	1.0	0.0	
2	1.0	0.0	0.0	
3	0.0	0.0	0.0	
4	0.0	0.0	1.0	
...	...	...	...	
795	0.0	0.0	0.0	
796	0.0	1.0	0.0	
797	0.0	1.0	0.0	
798	0.0	0.0	0.0	
799	1.0	0.0	0.0	

800 rows × 31 columns

Fig. 8. One-hot-encoding

Source and Destination: In the sourceand destination part along with whether the airfline is travelling in the northern part or the southern part. The airline has been aggregated in these manner. For the source and destination we have used the mean encoder along with the Power transformer to get better numerical representaion of categorical data.is both in

the northern region the price will be less as compare to if the source and destination are at different end of the country. Below is the screenshot of the representation:

	source	destination	source_is_north	destination_is_north
0	-0.060693	-0.075773	1	0
1	-1.144764	-1.751192	0	1
2	1.011077	1.011433	1	0
3	1.011077	1.011433	1	0
4	-0.060693	-0.075773	1	0
...	...	...	...	...
795	1.011077	1.011433	1	0
796	-1.717174	-1.074298	1	0
797	-1.717174	-1.074298	0	1
798	1.011077	1.011433	1	0
799	-0.060693	-0.075773	1	0

Fig. 9. Representation of Label Encoder

It was even performed different time segment in which the travelling is been done at which time of the day, whether its morning,afternoon and evening.This method was implemented using the data time features which segregate the days in different times and used the pipeline method from sklearn to get the better numerical representation. Following the Image output gives better clarity as how the data is represented numerically:

	dep_time_hour	dep_time_minute	arrival_time_hour	arrival_time_minute	dep_time_part_of_day	arrival_time_part_of_day
0	0.391304	0.909091	0.521739	0.545455	1.000000	0.000000
1	0.913043	0.272727	0.000000	0.272727	0.152985	1.000000
2	0.782609	0.818182	0.826087	0.272727	0.220149	0.755906
3	0.782609	0.000000	0.043478	0.545455	0.220149	1.000000
4	0.869565	0.454545	0.913043	0.090909	0.152985	1.000000
...	...	...	...	...	...	...
795	0.347826	0.000000	0.652174	0.545455	1.000000	0.000000
796	0.391304	0.181818	0.434783	0.727273	1.000000	0.921260
797	0.956522	0.090909	0.000000	0.454545	0.152985	1.000000
798	0.173913	1.000000	0.826087	0.000000	1.000000	0.755906

Fig. 10. Arrival-Depart time representation

After the implementation of the Feature engineering technique where the complete dataset have been completed into the required numerical representaion. The total input features comes to be 31 columns. The random regressor classifier has been implemented which works as a feature selection. This helps to reduce the input features from 31 columns to 9 columns. The image shows the output size of the featured model to get the details.

III. Experiments and Results:

For experimentation and evaluating the dataset that we have gotten to implement different ML algortihms and done the hyperprarameter tuning to check as which model performs the best out of those models used.

A. Implemented the Linear Regression model

Linear regression is a fundamental statistical method used to model the relationship between a dependent variable and

	air__airline_Indigo	air__airline_Jet Airways	air__airline_Other	location__source	location__destination	dur__duration_cat	dur__duration	stops__total_stops	stops__is_direc
0	0.0	0.0	0.0	-0.060693	-0.075773	2.0	1.888583	2.0	
1	1.0	0.0	0.0	-1.144764	-1.751192	1.0	-0.934723	0.0	
2	0.0	0.0	0.0	1.011077	1.011433	2.0	1.630112	2.0	
3	0.0	0.0	0.0	1.011077	1.011433	2.0	-0.397897	1.0	
4	0.0	1.0	0.0	-0.060693	-0.075773	2.0	1.649994	1.0	
...	...	...	...	...	...	...	...	...	
795	0.0	0.0	0.0	1.011077	1.011433	2.0	-0.397897	1.0	
796	1.0	0.0	0.0	-1.717174	-1.074298	0.0	-1.113665	0.0	
797	1.0	0.0	0.0	-1.717174	-1.074298	0.0	-1.014253	0.0	
798	0.0	0.0	0.0	1.011077	1.011433	2.0	0.387459	1.0	
799	0.0	0.0	0.0	-0.060693	-0.075773	2.0	0.288047	1.0	

800 rows × 9 columns

Fig. 11. Featured Data Output

one or more independent variables. It aims to find the best-fitting straight line (the regression line) through the data points, which minimizes the sum of squared differences between the observed values and the predicted values. It is implemented using the sklearn library. The loss function used here are the MSE(Mean Squared Error),MAE(Mean Absolute Error) and the R2 score.

```
from sklearn.linear_model import LinearRegression
#XX_train, XX_test, yy_train, yy_test = train_test_split(X_PRE, Y, test_size=0.2, random_state=42)

# Initialize Linear Regression model
linear_model = LinearRegression()

# Train the Linear Regression model
linear_model.fit(XX_train, yy_train)

# Make predictions on the test set
y_pred_linear = linear_model.predict(XX_test)

# Evaluate the Linear Regression model
mse_linear = mean_squared_error(yy_test, y_pred_linear)
mae_linear = mean_absolute_error(yy_test, y_pred_linear)
r2_linear = r2_score(yy_test, y_pred_linear)

# prinitng the losses and the evaluation metric to compare the results of different algorithms
print(f"Linear Regression - Mean Squared Error (MSE): {mse_linear:.2f}")
print(f"Linear Regression - Mean Absolute Error (MAE): {mae_linear:.2f}")
print(f"Linear Regression - R-squared (R2): {r2_linear:.2f}")
```

```
Linear Regression - Mean Squared Error (MSE): 10949066.45
Linear Regression - Mean Absolute Error (MAE): 2119.39
Linear Regression - R-squared (R2): 0.51
```

Fig. 12. Linear Regression Implementation

B. Gradient Boosting

Gradient Boosting is a powerful machine learning technique used for regression and classification tasks. It builds models sequentially by combining the predictions of multiple weak learners, typically decision trees, to create a strong predictive model. Each new model in the sequence is trained to correct the errors of its predecessor by focusing on the residuals, or differences between the actual and predicted values. The process involves minimizing a loss function using gradient descent, hence the name ”Gradient Boosting.” This method is highly effective in reducing bias and variance, leading to models with excellent predictive performance.

C. Bagging

Bagging, short for Bootstrap Aggregating, is an ensemble learning technique designed to improve the stability and accuracy of machine learning models. It works by creating multiple subsets of the original dataset through random sampling with replacement. Each subset is then used to train a separate model, typically decision trees. The final prediction is made by averaging the predictions (for regression) or taking a majority

vote (for classification) from all the models. Bagging reduces variance and helps prevent overfitting, making it particularly effective for high-variance models like decision trees.

D. XGBoost

XGBoost, or Extreme Gradient Boosting, is an advanced and efficient implementation of the gradient boosting algorithm. It enhances gradient boosting by incorporating techniques like regularization (to reduce overfitting), parallel processing (for faster computations), and tree pruning (to optimize model complexity). XGBoost builds models sequentially, where each new model corrects the errors of the previous ones by focusing on the residuals. It is highly effective in handling large datasets and complex tasks, making it a popular choice in competitive machine learning due to its speed, accuracy, and scalability.

```
# Perform Grid Search
grid_search = GridSearchCV(estimator=xgboost_model, param_grid=param_grid, scoring='neg_mean_squared_error', cv=3, verbose=1)
grid_search.fit(XX_train, yy_train)

# Get the best parameters
best_params = grid_search.best_params_
print("Best parameters found: ", best_params)

# Train the model with the best parameters
xgboost_best_model = xgb.XGBRegressor(**best_params)
xgboost_best_model.fit(XX_train, yy_train)

# Evaluate the model
y_pred_best = xgboost_best_model.predict(XX_test)
mse_best = mean_squared_error(yy_test, y_pred_best)
mae_best = mean_absolute_error(yy_test, y_pred_best)
r2_best = r2_score(yy_test, y_pred_best)

print(f"XGBoost Best Model - MSE: {mse_best:.2f}")
print(f"XGBoost Best Model - MAE: {mae_best:.2f}")
print(f"XGBoost Best Model - R2: {r2_best:.2f}")
```

```
[2] Requirement already satisfied: joblib in c:\users\singh\anaconda3\envs\efficientps_env\lib\site-packages (1.3.2)
Note: you may need to restart the kernel to use updated packages.
Fitting 3 folds for each of 324 candidates, totalling 972 fits
Best parameters found:  {'colsample_bytree': 0.7, 'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 100, 'subsample': 0.8}
XGBoost Best Model - MSE: 5789902.67
XGBoost Best Model - MAE: 1444.96
XGBoost Best Model - R2: 0.74
Fitting 3 folds for each of 324 candidates, totalling 972 fits
Best parameters found:  {'colsample_bytree': 0.7, 'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 100, 'subsample': 0.8}
XGBoost Best Model - MSE: 5789902.67
XGBoost Best Model - MAE: 1444.96
XGBoost Best Model - R2: 0.74
```

Fig. 13. XGBoost Implementation

SQ is evaluating how closely our segments are with their ground truths. When it comes to value 1, it means our TP predicted segments are more closely matched with their ground truths. It doesn't take into account any of our bad predictions. RQ measures the ability of the model to predict the classes accurately.
The plot for Panoptic Quality is shown in figure 7. The maximum PQ achieved is 44.5. The blur line shows the actual result and highlighted line shows the result after smoothing.
The plot for Segmentation Quality is shown in figure 8. The maximum SQ achieved is 75.9.
The plot for Panoptic Quality is shown in figure 9. The maximum RQ achieved is 56.8.

IV. Conclusion

Out of all the algorithms that has been implemented the XGBoost gives the best result. I have implemented the XGboost model on the streamlit to get the final price prediction. Shown is the summary of the algoithms:

```
Model Comparison:
              Model           MSE         MAE        R2
0  Linear Regression  1.094907e+07  2119.394628  0.508200
1           XGBoost  5.919473e+06  1446.896110  0.734115
2           Bagging  6.082269e+06  1487.763696  0.726802
3  GradientBoosting  6.096497e+06  1607.889156  0.726163
```

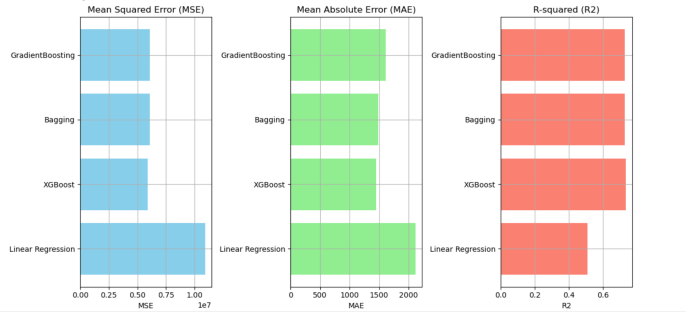

Fig. 14. Final Results

References

[1] [1603.02754] XGBoost: A Scalable Tree Boosting System - arXiv.org
[2] https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
[3] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 2980-2988, doi: 10.1109/ICCV.2017.322.
[4] https://feature-engine.trainindata.com/en/latest/.
[5] [1908.06951] Gradient Boosting Machine: A Survey - arXiv.org
[6] Xiong, Y., Liao, R., Zhao, H., Hu, R., Bai, M., Yumer, E., & Urtasun,R. (2019b). Upsnet: A unified panoptic segmentation network. In Proceedings of the conference on computer vision and pattern recognition (pp. 8818–8826).
[7] Jan Hosang, Rodrigo Benenson, Bernt Schiele Learning non-maximum suppression in CVPR , 2017